LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-587533

Characterizing the behavior of bandwidth-bound applications on torus networks

N. Jain, A. Bhatele, H. Menon, T. Gamblin, M. Schulz, L. V. Kale

October 3, 2012

27th IEEE International Parallel & Distributed Processing Symposium
Boston, MA, United States
May 20, 2013 through May 24, 2013

Disclaimer

Characterizing the behavior of bandwidth-bound applications on torus networks

Nikhil Jain*, Abhinav Bhatele†, Harshitha Menon*, Todd Gamblin†, Martin Schulz† and Laxmikant V. Kale*

*Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA
†Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551 USA
E-mail: {nikhil, gplkrsh2, kale}@illinois.edu, {bhatele, tgamblin, schulzm}@llnl.gov

***Abstract*—It is well known that the communication performance of parallel applications can depend heavily on the mapping of tasks or processes to physical nodes in the interconnection network. Much prior work has focused on optimizing latency-bound algorithms by placing frequently communicating tasks near each other on the network. However, latency optimizations do not necessarily help bandwidth-bound applications. To optimize bandwidth, we must maximize the number of links used to transfer message packets on the network without increasing the number of messages contending for the same links. This tradeoff is not well understood.**

In this paper, we develop a methodology to correlate network mappings with the contention they create as well as their impact on application performance. We develop a novel metric, called plateau point, that ascribes a single value to the network counter values for an application run, and we show that this metric is correlated to communication performance in bandwidth-bound applications. This metric will enable predicting application performance without doing actual runs or simulations as long as we have a model for routing of messages through the network. We also describe the queuing strategy on Blue Gene machines and the impact of the default settings on the performance of large messages. Our insights can be applied by network designers when optimizing for bandwidth-bound applications. Our study of task mappings and trade-offs between maximizing effective bandwidth and minimizing network contention is a first step towards creating automated mappings for bandwidth-bound applications.

***Keywords*-task mapping, network contention, hardware counters, performance prediction, correlation**

I. Introduction

The task mapping problem, i.e., how to optimally map a set of processes onto a communication network, is of fundamental importance to high performance computing because the networks used on today's largest machines do not have flat topologies. To reduce costs and to allow high scalability, the largest supercomputers have adopted tori, meshes, and other high diameter networks. If application tasks are laid out poorly on these topologies, performance can degrade due to high latency, high contention, or low numbers of available network links.

Much work has been done to increase the performance of latency-bound applications running on these networks. Typically, heuristic algorithms attempt to lay out frequently communicating tasks so that the number of network hops between the tasks is minimized. Bandwidth optimization is fundamentally different. Rather than a clear minimization problem, bandwidth mapping requires that we maximize the number of routes available between communicating processes. We can increase the number of available routes by increasing the number of dimensions between communicating tasks in n-dimensional Cartesian networks [8]. This allows for more bandwidth to be injected into the network through the new routes, but this will also increase traffic on the links and thereby increase the potential for contention.

The tradeoff between bandwidth and contention is not well understood, and this prevents us from reliably predicting the performance effects of bandwidth-optimizing task mappings on the largest supercomputer networks. Several obstacles prevent a full understanding. First, contention is fundamentally a dynamic property that depends on the routing behavior of the network, the layout of tasks, and the communication schedule of the parallel application. Typical schemes to predict contention rely on low-level network simulation [19], [14], [17]. Second, parallel applications themselves are complex and dynamic, and they may contain many different non-overlapping and overlapping communication phases. We cannot rely entirely on aggregate, whole-run measurements to predict contention. Finally, while we may be able to position tasks to increase available routes between two nodes, the network may not allow us to inject bandwidth onto all routes concurrently, reducing the total available bandwidth. We have discovered that even on networks that do support high-bandwidth injection, there may be limitations that prevent high speed injection of certain types of traffic, again making mapping performance difficult to predict.

In prior work, we introduced primitive task mapping transformations that can increase the available bandwidth in parallel applications [8]. Our goal is to come up with a set of rules and heuristics that will enable the generation of automatic bandwidth-optimizing mappings for parallel codes. To accomplish this, we need concise and effective metrics to accurately estimate the contention/bandwidth tradeoff, so that we can use these metrics to guide optimization algorithms. This paper offers the following contributions:

1) We develop a methodology to correlate task mappings with statistical characteristics of the network traffic they create and its impact on application performance.
2) We develop a novel metric, called plateau point, that

ascribes a single value to the network counter values for an application run, and we show that this metric is correlated to communication performance in bandwidth-bound applications.

3) We also describe the queuing strategy on Blue Gene machines and the impact of the default settings on the performance of large messages. Our insights will be useful to network designers when optimizing for bandwidth-bound applications.

The plateau point will enable predicting application performance without doing actual runs or simulations as long as we have a model for routing of messages through the network. Our study of task mappings and trade-offs between maximizing effective bandwidth and minimizing network contention is a first step towards creating automated mappings for bandwidth-bound applications. We anticipate that it will enable us to develop predictive heuristics and algorithms in future work.

The remainder of this paper is organized as follows: Section II presents related work. In Section III, we describe our methodology for correlating network counters with application communication performance. There are important steps in this process, which are explained in detail in Sections IV, V and VI. In Section VII, we show the application of our methodology to different bandwidth-bound applications and conclude the paper in Section VIII.

II. Related Work

Performance analysis of communication patterns and contention on large scale systems has been done in several studies using either theoretical models, empirical results or a combination of the two. Studies based on theoretical models such as LogP [10], LogGP [3], LoPC [12] and LoGPC [15] have proved useful in understanding generic network characteristics. However, these models are not accurate for cases with non-uniform traffic distribution over the network. Non-uniform traffic distribution is commonly found in most dynamic application executions and can be caused by several factors such as non-uniform communication patterns, the network's routing algorithm or limited link bandwidth. On the other hand, empirical studies [2], [18], [9] have focused on presenting a large set of results with variation in system configuration, communication volume and mapping. They also provide a generic reasoning for the observed performance based on the understanding of the specific communication patterns and observed behavior of the networks, but do not provide an in-depth analysis.

This paper presents an analysis technique that belongs to the category of empirical studies, but performs a detailed analysis. We propose a metric based on the flow of traffic on the network, that correlates with the observed performance. Other metrics have been proposed previously to predict the performance of task mapping for a given communication pattern. Several studies [11], [1], [6] have shown that hop-bytes or average hops per byte correlates well with overall performance in presence of contention. Hoefler et al. [13] propose the use of communication volume on individual links as an optimization metric to design mappings that improve performance. Balaji et al. [5] have studied the correlation of NO_TOKEN counters with execution time of global communication patterns on Blue Gene/P. Bhatele et al. [7] have shown empirically that the total communication traffic correlates well with the overall performance for some applications.

III. Methodology to Understand Application Communication Performance

The effect of task mapping and routing algorithms on the resulting contention on interconnection networks and the impact on application performance is not well understood. Application developers often collect hardware counter data for the network [7]; however, there is no clear metric correlating network counter data to actual performance. In this section, we outline a process of correlating link counter data, obtained on Blue Gene machines, with actual application performance. This method can help us develop models for predicting application performance given an application communication graph, its mapping on the network and the routing algorithm for messages.

The High Performance Monitor (HPM) library on Blue Gene machines provides network counters such as the number of packets passing through each link (`BGP_TORUS_XP_PACKETS` etc., for the six links on each node) and the number of times the next node, that a given node is trying to forward a message to on an available link, has no buffers to accept the message (`BGP_TORUS_XP_NO_TOKENS` etc., for the six links). We use link counters that give the number of packets passing through each network link to correlate network traffic with the application performance. Our hypothesis is that when comparing two different executions of an application, if less traffic (fewer packets) pass through most links on the network in the first execution than the second, then the performance is better in the first execution.

We can use an empirical probability density function (PDF) or an empirical cumulative distribution function (CDF) to plot link counters data and correlate it with performance. Figure 1 shows an *empirical* PDF (top) and CDF (bottom) for network counter data obtained for two very different node mappings applied to pF3D (a multi-physics production application at LLNL, which is introduced in more detail in Section VII-B) on 32,768 cores of Blue Gene/P. The x-axis on both plots has histogram bins that represent the number of packets passing through each network link (in X, Y or Z direction). The y-axis in the PDF plot shows the fraction (between 0 and 1) of links that fall under a particular bin. Hence it shows the distribution of the number of packets flowing through different links on the network. In the CDF plot, the y-axis shows the cumulative fraction of links that send number of packets less than or equal to the particular bin.

The PDF in Figure 1 shows that Map 1 has about 15% of links in two bins towards the right that represent around 2 and 3 million packets respectively. Map 2 has these links in significantly smaller bins on the left, around 0.2 to 0.4

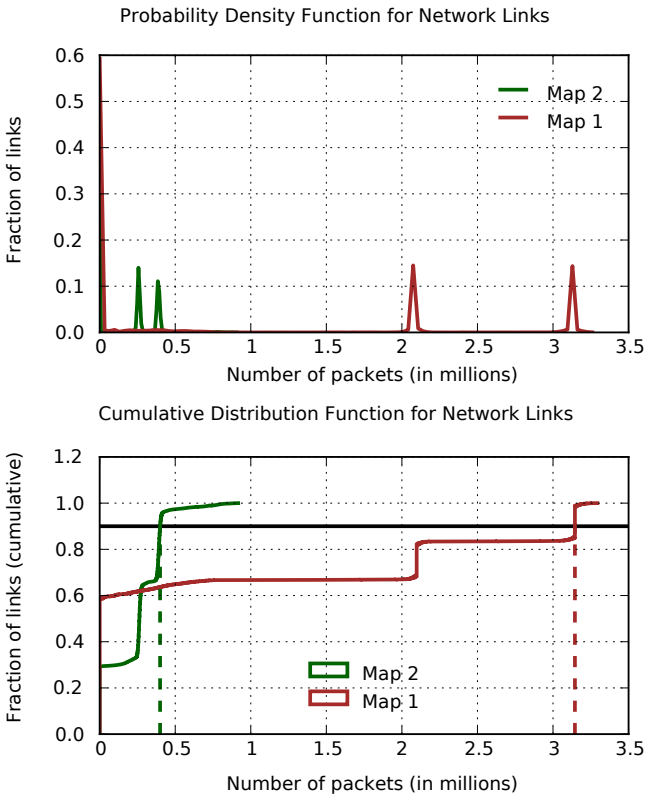


Fig. 1: Empirical PDF and CDF of number of packets sent on each network link for two different mappings of pF3D. The x-axis on both plots has histogram bins that represent the number of packets passing through each network link (in X, Y or Z direction)

million packets. If we look at the same information plotted as a CDF, it is more obvious that Map 2 approaches the maximal plateau (of 100% of the links sending less than a certain number of packets) much earlier than Map 2. In order to capture this notion of when the plateau is reached for a given mapping/execution, we introduce a metric called the "plateau point". The plateau point is a point (x, y) on the CDF curve and denotes that at this point, y% of all links on the network send fewer than x number of packets for the profiled region, with y being 0.9 (90%) in the example shown in the figure. By varying y, we can identify a point on the CDF curve where most of the links sent a bounded number of packets. We use this bound to approximate that position in CDF after which the CDF flattens. If chosen properly, this bound guarantees that beyond this point, there will be no steep rise in the CDF, and hence there will not be any heavily loaded bins to its right. Our hypothesis is that this bound correlates well with actual application performance. If the bound is low, there is a smaller chance of contention and hence better communication performance.

There is a three-step process to obtain good correlations between mappings/network counter values and communication performance using the plateau point. The steps are introduced below and explained in subsequent sections in more detail:

Step I. Remove injection bottlenecks: In the process of creating bandwidth bound mappings, we discovered that adding more routes for message packets can be futile if the scheme for injecting messages from the processor onto the links is not optimal. If the messages for a given destination can use only one of many queues (FIFOs) for injecting messages, then increasing the number of available routes for bandwidth optimization may not improve performance. As a consequence, the network counters might tell a wrong story because the bottleneck is not on the links but elsewhere. Hence, it is important for bandwidth-optimizing mappings to minimize injection bottlenecks first.

Step II. Breakdown into non-overlapping phases: If an application with several non-overlapping phases is profiled from start to end, the network counter values represent the total network traffic over a span of time. It is possible for a link to have a very high counter value but its traffic might be spread over time. Hence, it is extremely important to profile "small" code regions within the application that are independent and in which all communication happens simultaneously.

Step III. Decide the plateau point: Assuming that we follow the two conditions in step I and II, we can use the plateau point metric to correlate network counter values with performance. There is still the question of how do we choose a good value, y for the plateau point (x, y). This will be explained in Section VI.

IV. Step I. Remove Injection Bottlenecks

On Blue Gene/P, messages are injected onto the network using a direct memory access (DMA) engine. To send a message, a core enqueues a matching message header in one of the injection FIFOs available within the processor. By default, 6 injection FIFOs are created for messages going out of the node. The selection of which FIFO to inject the message header into is done by the core based on the destination node (Figure 2). The DMA engine inspects the processor injection FIFOs and copies message data from memory to the torus injection FIFOs from which it is sent onto the torus links. In the default setting, there is a one-to-one mapping between the processor injection FIFOs and the torus injection FIFOs, i.e., packets created for a message whose header was inserted into a particular processor injection FIFO can only be injected into the corresponding torus injection FIFO. Hence, in this setting, a message can be injected onto the torus links from one torus injection FIFO only.

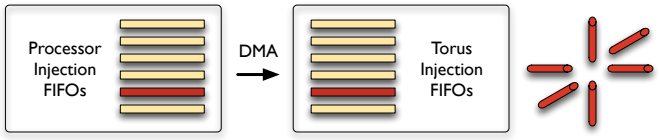


Fig. 2: Default one-to-one mapping of processor injection FIFOs to torus injection FIFOs.

Bandwidth optimizations require placing communicating tasks on diagonally opposite corners of a cube in order to

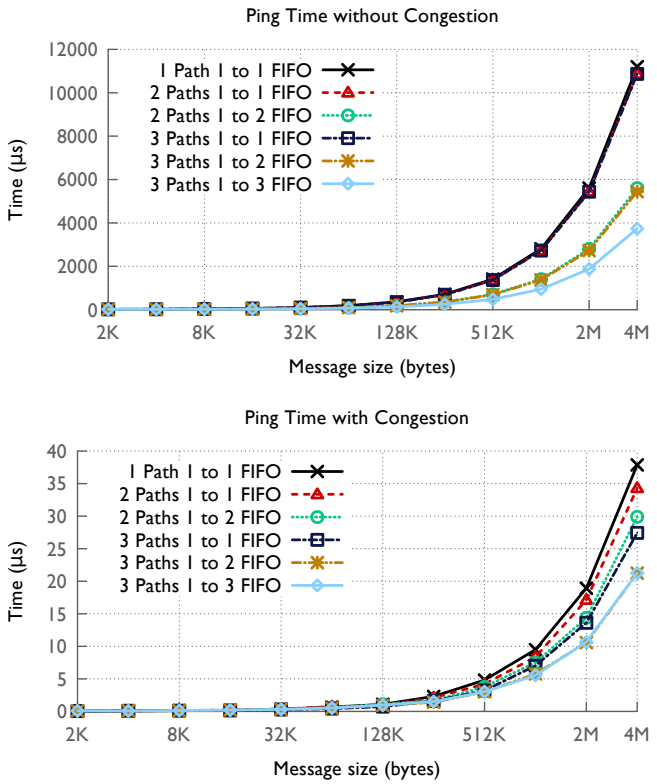


Fig. 3: Performance of a one-sided ping for different processor pairs and different mappings of the processor to torus injection FIFOs

provide multiple paths for the packets to be routed through. However, this cannot be exploited if the rate of injecting packets is limited to one torus FIFO for one destination. We created a simple experiment to test this. We allocate a $8\times8\times8$ torus on Blue Gene/P and perform `MPI_Sends` between a pair of processes. The processes are placed either 1 hop away (1 path between them) or 2 hops away on the corners of a square (2 disjoint paths between them) or 3 hops away on the corner of a cube (3 disjoint paths between them). For the default one-to-one mapping of processor to torus injection FIFOs, we see no performance difference in the three cases (first three curves in Figure 3, top). If we look at the traffic flow for the 3 hops/paths case, it looks like the left image in Figure 4. So even though the network traffic appears to be using three different paths, we do not see any performance improvement as compared to the 1 hop/path case. This can be explained by the default mapping of FIFO queues causing the traffic being sent along different routes to be not overlapped.

We then modified the default mapping between the two sets of FIFOs. We tried a 1 processor FIFO to 2 torus FIFOs mapping and a 1 processor FIFO to 3 torus FIFOs mapping in addition to the default. We can see that for the 1 to 2 FIFOs mapping, the performance of the ping improves, by two times, when placing the processes on a square (2 paths) or cube (3 paths). The performance gets even better when we do a 1 to 3 processor to torus FIFO mapping and place the communicating processes on a cube. If we look at the network traffic for this

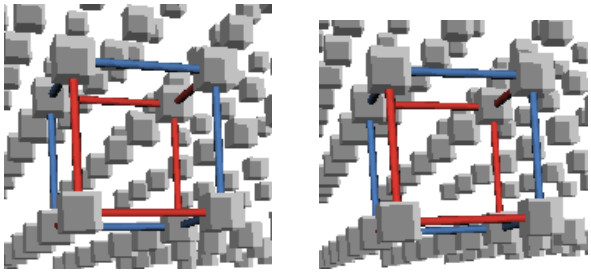


Fig. 4: Visualizations of the network traffic between a pair of processes placed on the corners of a cube: 1 to 1 FIFO mapping (left) and 1 to 3 FIFO mapping (right) using Boxfish

last case, it looks exactly the same as that for a 1 to 1 FIFO mapping (comparing left and right images in Figure 4). The only difference is that in the image in the right, the packets are being sent on the network in parallel on various links.

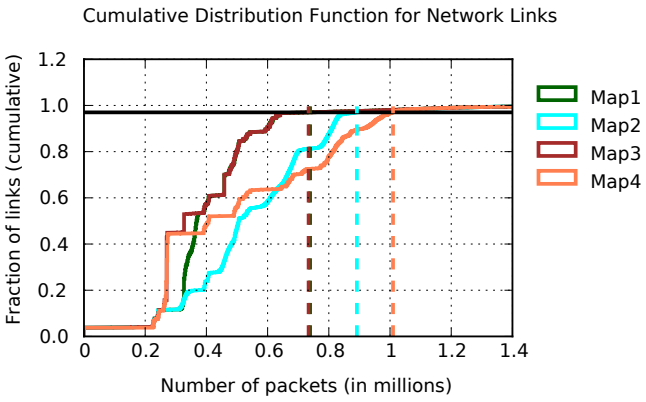
These results suggest that it is important to have a custom setting for the mapping of processor to torus injection FIFOs to best utilize the bandwidth in multiple dimensions. Even better will be scenario if this mapping can be decided on per message basis instead of being attached for the FIFOs. The network views that shows the the traffic suggest that it is important to separate out the packet counter values for different non-overlapping phases for comparison. In absence of such a breakdown, the counters are of limited used. The bottom plot in Figure 3 shows results similar to the top figure but with an all-to-all communication happening in parallel on all other processes to create background contention. The results are similar to the case with no background traffic although the improvements from custom FIFO/queue mappings are smaller. This suggest that for an application with every processor communicating with a large number of processors, and hence causing congestion on the network, these FIFO mappings may not impact performance. However, in communication patterns where the number of neighbors is limited, performing a custom FIFO mapping may improve performance.

V. Step II. Breakdown into Non-overlapping Phases

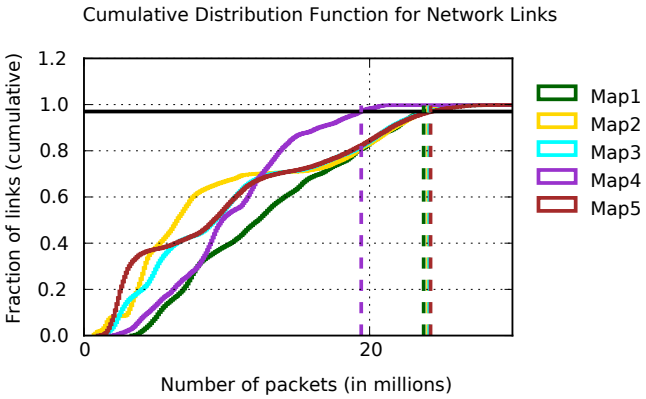
If an application with several non-overlapping phases is profiled from start to end, the network counter values represent the total network traffic over a span of time. It is possible for a link to have a very high counter value but its traffic might be spread over time. Hence, it is extremely important to profile "small" code regions within the application that are independent and in which all communication happens simultaneously.

In this section, we present results for end-to-end runs of pF3D and NAS-CG (Section VII) – counters and timers are activated when the application begins execution and are deactivated when the application terminates. Based on these results, we motivate the need for per-phase breakdown of counters and timers to accurately correlate observed performance with the proposed metric.

Figure 5a presents the CDF plots with plateau point at



(a) pF3D - Aggregate CDF



(b) NAS-CG - Aggregate CDF

Fig. 5: CDF and the plateau point for aggregated counters for the entire run show weak correlation

$(x, 0.97)$ for a pF3D run on 8,192 nodes (32,768 cores) of Blue Gene/P. Four different tasks mappings, Map1, Map2, Map3 and Map4, whose execution time increases in the same order, i.e., $T_{Map1} < T_{Map2} < T_{Map3} < T_{Map4}$ (Table I), were used in this experiment. It can be seen that, using plateau point with $y = 0.97$, the prediction based on x value of plateau point is not accurate. Even after trying a range of y values for the plateau point, we did not find any y value for which the prediction matches the observed performance. Similar results are observed for NAS-CG as shown in Fig. 5b. There is approximately 10% performance difference between the best predicted mapping (Map4) and the best observed mapping (Map1). However, note that for both applications, the prediction of relative performance of many other mappings was found to be correct.

From these experiments we conclude that there is a possible correlation between the the x values of plateau points (with a good y value) and the performance of an application, but the accuracy of these predictions is questionable for some cases. This conclusion brings the following question to one's attention - did some information get lost due to aggregation of counters over the end-to-end runs? In the following section, we attempt to answer this question by performing analysis

Mapping	pF3D	CG
Map1	94.98	28.57
Map2	95.82	29.27
Map3	98.39	31.94
Map4	105.5	31.95
Map5	–	32.41

TABLE I: Aggregate time (in seconds) for pF3D and CG

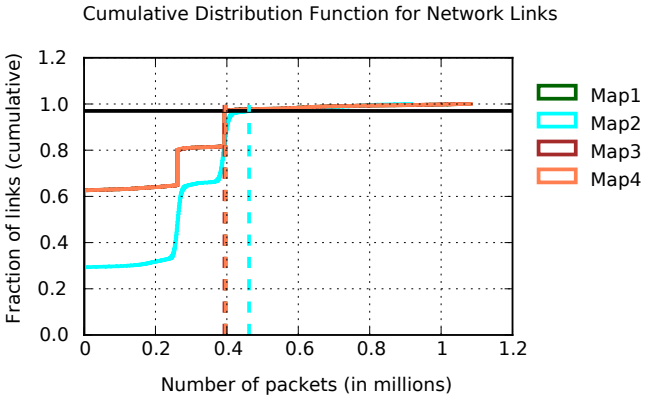


Fig. 6: CDF for one phase of pF3D: accuracy improves when plateau point based prediction is done for one single phase

of only one phase of communication of pF3D that does not overlap with any other phase.

A. *Per-phase Analysis*

Building on the intuition from the previous section, we divided pF3D into distinct communication phases (§VII-B), and analyzed results for them. In Figure 6, we present CDF plots for one such phase, and analyze its correlation with the observed performance. We stick to the $y = 0.97$ value for plateau point used for the aggregated case for these results. Predicting based on the x values of plateau points in Figure 6, the timing for Map1, Map3 and Map4 for this phase should be similar. The performance of Map2 should be close, but slightly worse, given that x value of its plateau point is to the right of the x value of the plateau point of other mappings. The observed time for this phase of pF3D is shown in Table II. It can be seen that the predictions done for this phase are closer to the observed performance in comparison with the aggregated results. However, there is still room for improvement, and the predictions need to be refined further. In the next section, we explore the solution space for the best suited y value, and revisit per-phase analysis in Section VII.

VI. Step III. Decide the plateau point

The plateau point metric can be used to correlate network counter values with performance. However, we need to choose a good value, y for the plateau point (x, y). To recap, at this point, y% of all links on the network send fewer than x number of packets for the profiled region. By varying y, we can identify a point on the CDF curve where most of the links send a

Mapping	Time (ms)
Map1	45.31
Map2	45.96
Map3	44.92
Map4	45.18

TABLE II: pF3D time for one phase

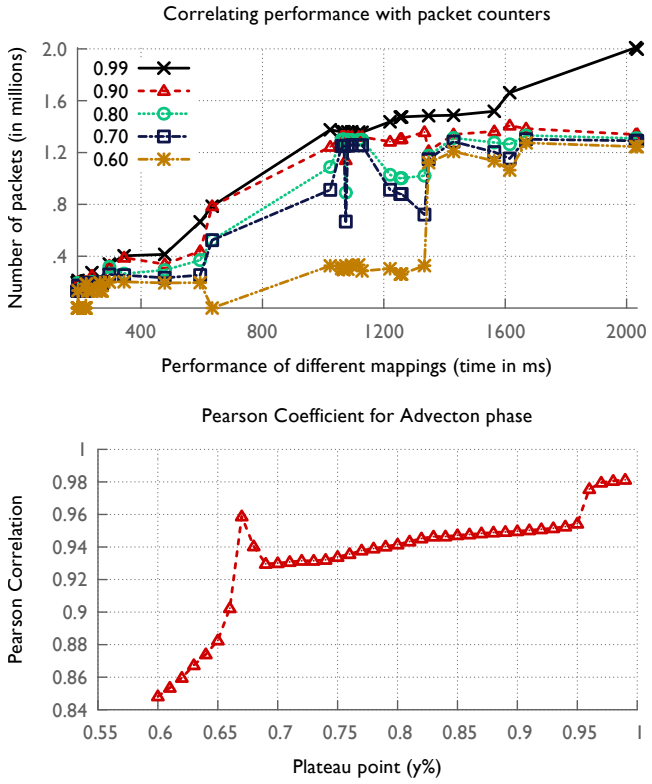


Fig. 7: Correlating performance and the plateau point by varying y: Advection phase of pF3D

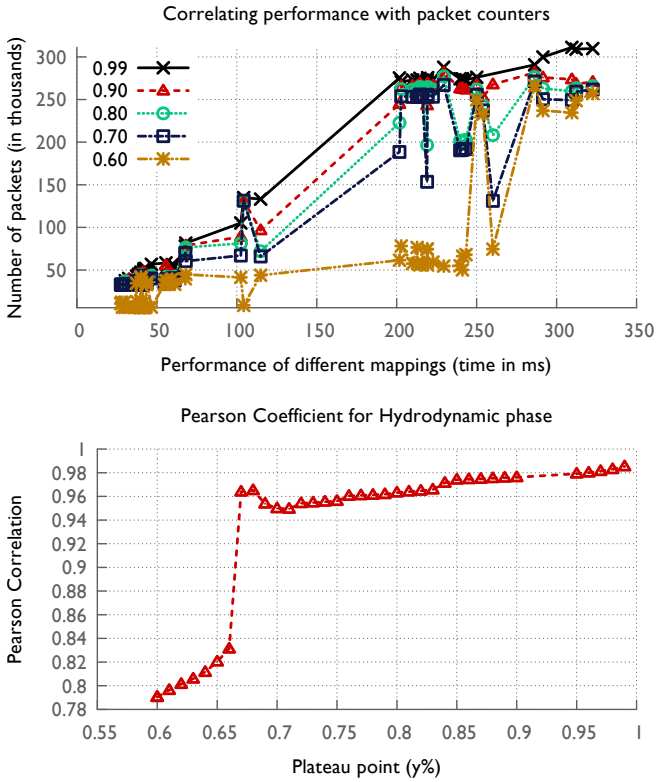


Fig. 8: Correlating performance and the plateau point by varying y: Hydrodynamics phase of pF3D

bounded number of packets. We use this bound to approximate the position in CDF after which it flattens. If chosen properly, this bound guarantees that beyond this point, there will be no steep rise in the CDF, and hence there will not be any heavily loaded bins to its right.

It is important to find a good value for the plateau point (y%) to ensure that it correlates with the communication characteristics well. For this purpose, we generated a large set of task mappings (60+) for pF3D (Section VII-B), and experimented with a y value of $[0.60, 1.00]$.

Figure 7 presents the correlation of performance (execution time on the x-axis) with the x value of the plateau point (on the y-axis) for different values of y (0.6, 0.7, 0.8, 0.9 and 0.99). This is for one of the phases of pF3D. Given a value of y, for each of the task mappings, we add a point to the plot based on its performance and the x value of the plateau point. In Figure 7, it can be seen that when y is small, the correlation is weak: there are many cases in which a mapping performs better than the other mappings even though its x value (number of packets) is larger than that of other mappings. Similar observation can be made for another phase of pF3D shown in Figure 8. However, in both phases, for the best choice of $y = 0.99$, we observe a strong correlation between the performance and the x values (represented by the enveloping black curve at the top). This strong correlation is supported by the high value of Pearson Coefficient which we obtain for these runs as shown in the figures. For both the phases, we find that as the x or the number of packets increases, the performance becomes worse. This suggests that, for the applications used in this paper, a plateau point with $y = 0.99$ is a good value for finding the bin at which CDF reaches its plateau. Hence, in rest of this paper, we will use $y = 0.99$ as our plateau point for our analysis.

It may appear from the above results that one may be able to correlate performance of a mapping with the number of packets sent on the most loaded link, and need not look for a plateau point. This corresponds to using a value of $y = 1.0$ as the plateau point. In Figure 9, we present the correlation of performance of NAS-CG with the x values of the plateau point (Section VII-C) with y set at 1.0. It can be seen in the given plot that there is little correlation between the packets on the most loaded link and the observed performance. In a later section (Section VII-C), we plot a similar graph for plateau point calculated using $y = 0.99$, and show a strong correlation with the performance.

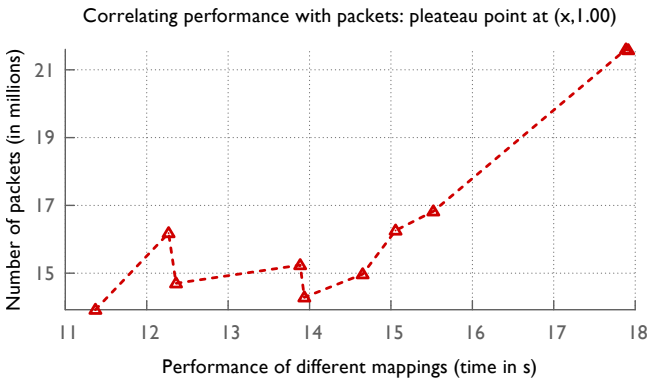
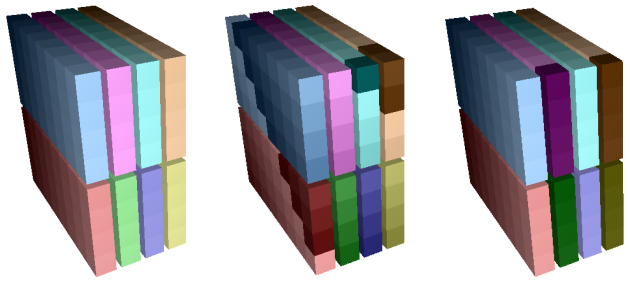


Fig. 9: Correlating performance with packets: Why picking y=1.0 for the plateau point is not good?



(a) Tile operation (b) Tilt operation (c) Zigzag operation

Fig. 10: Operations in Rubik: tile, tilt, zigzag

VII. Correlating Application Performance to Network Counters

A. Task Mappings

We use a number of mapping schemes to map the logical topology of processes (linear, grids, etc.) to the physical 3D-torus of Blue Gene/P. Except for the default rank order mappings available on Blue Gene/P, we use Rubik [8], a tool that enables user to describe variations in task mappings using a small domain specific language, to generate all other task mappings. The following operations, provided by Rubik, have been used to generate the task mappings:

Tile - This operation divides a parent space into fixed-size child spaces or tiles. For example, as shown in Figure 10a, a 3D-grid can be divided into 8 grids. In general, we have tiled the logical topology and the 3D-torus individually, and mapped them in a one-to-one manner. Such tiling is very useful when we have a set of processors, e.g. in a local all-to-all, and we wish to map them close to each other in a cube.

Tilt - We use tilt operations to shift planes of a topology normal to their direction in a successively increasing manner (Figure 10b). This operation is useful in increasing the number of available paths between neighboring processors.

Zigzag - Zigzag operation is used to apply fixed offset shift to alternating planes normal to the direction of the plane. An example shift operation is shown in Figure 10c.

B. pF3D

pF3D [16] is a multi-physics code from the National Ignition Facility at LLNL. It is used to study laser plasma-interactions in the experiments to predict the amount of scattering in proposed designs.

The simulated space in pF3D is a 3D-grid whose Z-direction is aligned with the laser beam. This 3D-grid is divided among a logical 3D-grid of processes. The simulation consists of three distinct phases: wave propagation and coupling, advecting light, and solving the hydrodynamic equations. Wave propagation and coupling consists of two-dimensional (2D) Fast Fourier Transforms (FFTs) in XY-planes; the 2D-FFTs are performed via two non-overlapping 1D-FFTs along X-lines and Y-lines using `MPI_Alltoall`. The advection phase involves ghost-plane exchange in the Z-direction performed using `MPI_Isend` and `MPI_Irecv`. Finally, the hydrodynamic phase consists of near-neighbor data exchange in the positive and negative X, Y and Z directions. The logical 3D-grid of processes used in this paper consists of 16 x 8 processes in XY planes, with the length of Z dimension calculated based on the total number of processes.

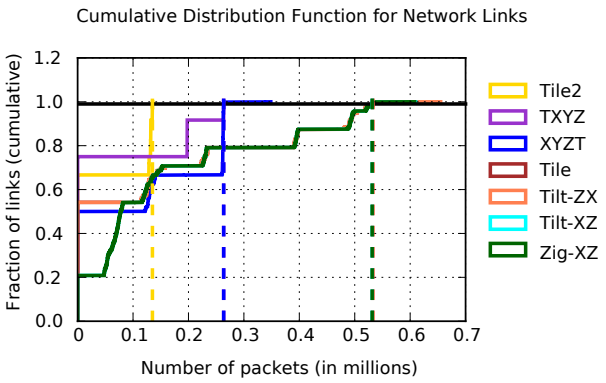
For pF3D, we use a set of seven mappings- two variations of dimension ordered mapping - TXYZ and XYZT, two types of tiling - Tile and Tile2, two tilted mapping based on Tile - Tilt-ZX and Tilt-XZ, and a zigzag mapping - Zig-XZ.

1) X-FFT and Y-FFT: pF3D consists of a 2D-FFT that is performed using two 1D-FFTs. These 1D-FFT's are performed using `MPI_Alltall` over sub-communicators created along X and Y dimensions. There is minimal overlap among the communication of the two 1D-FFTs, and hence we consider them as separate phases during counter collection.

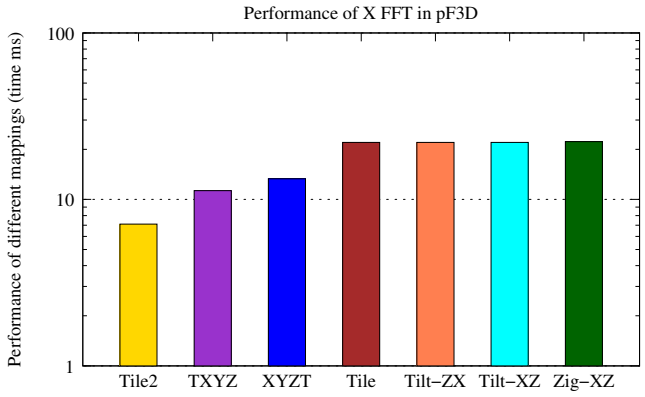
In Figure 11a, we present CDF plots and the plateau point of various mappings for X-FFT in pF3D. In the CDF plot, we see that Tile2 has the plateau point with lowest x value and hence, we expect its performance to be the best. Tile2 is followed by TXYZ and XYZT, who reach their plateau in close proximity, and are expected to have similar performance. Finally, the remaining set of mapping have almost identical x values of their plateau points, and we expect them to have matching performance. These predictions are confirmed by the time plot in Figure 11b where Tile2 outperforms all other mappings, and is followed by TXYZ and XYZT.

Another important thing to note here is the end point of CDF curves (which represent the maximum loaded link). We note that the end points of mappings in similar performing groups are far apart, and hence do not correlate with their performance.

Figure 12a shows the CDF plots for the Y-FFT phase. In contrast to X-FFT, more mappings have distinct CDF curves, and hence distinct plateau points. Moreover, based on correlation with x values of plateau points, a different set of mappings are expected to perform good. The set of three mapping Tile, Zig-XZ and TiltXZ, are expected to perform best followed closely by TXYZ. The other three mappings

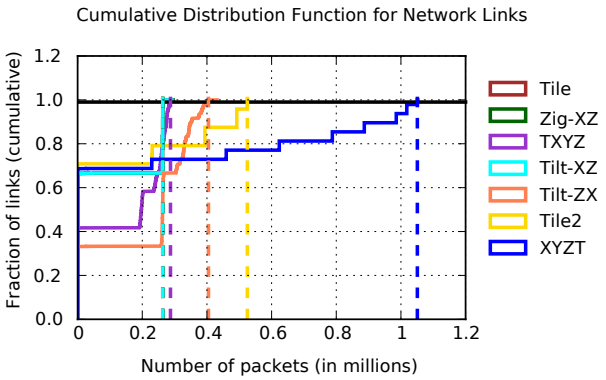


(a) CDF: Projection on x-axis predicts performance

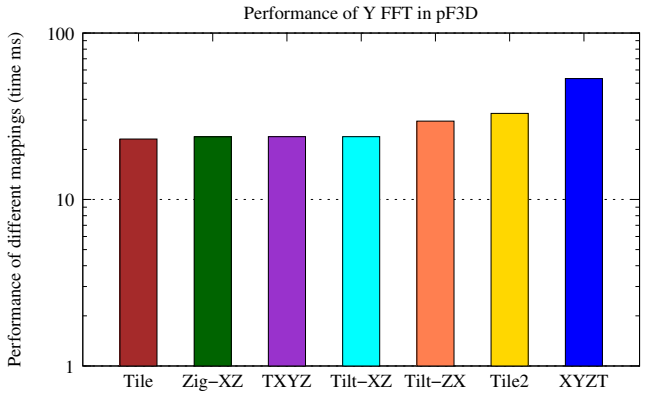


(b) Observed performance

Fig. 11: X-FFT using `MPI_Alltoll` over sub-communicators: x value of plateau point strongly correlates with the observed performance. As performance of a mapping gets worse, x value of plateau point moves towards right. Many mappings result in similar CDFs, and have overlapping curves.



(a) CDF: Projection on x-axis predicts performance



(b) Observed performance

Fig. 12: Y-FFT using MPI_Alltoall over sub-communicators: x value of plateau point correlates well with the performance. Note the difference from X-FFT in relative performance of mappings.

have much higher x values for their plateau points, and should perform significantly worse. A very similar trend is found in the time plot in Figure 12b. The set of three mappings Tilt-ZX, Tile2 and XYZT have very bad performance with XYZT being the worst among them. The other four mappings have an almost identical performance.

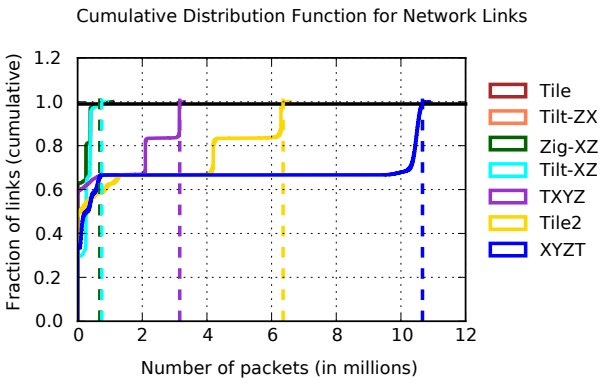
2) Advection: Ghost-Plane Exchange in Z: The Advection phase of pF3D requires information about adjacent Z planes to perform its computation. This results in ghost-plane exchange when the 3D-space is decomposed onto the 3D-grid of processes. The communication pattern is similar to ghost-exchange commonly seen in stencil computation.

We present the CDF curves for this phase in Figure 13a. Based on the plateau points shown in the CDF plot it is expected that the set of mappings - Tile, Tilt-ZX, Zig-XZ and Tilt-XZ - should have similar performance, and will outperform the other three mappings. The other three mappings, TXYZ, Tile2 and XYZT, with increasingly high x values for their plateau points should perform very badly. These predictions are confirmed by the timing results in Figure 13b in which we observe the group of four mappings with exactly same performance followed by the increasingly bad performance of the other three mappings.

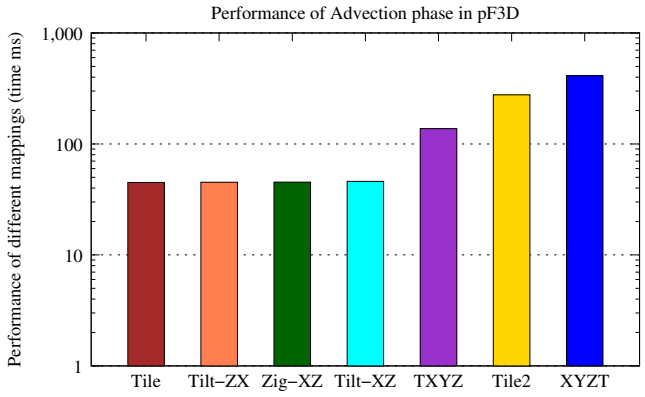
3) Hydrodynamic: Near-Neighbor Exchange in All Directions: The last phase of pF3D, hydrodynamic, involves a near-neighbor exchange in all six directions. Among the three phases of communication, this phase contains minimum communication volume. The CDF plots and associated plateau points for the near-neighbor exchange are presented in Figure 14a. The CDF plots for all mappings are very similar to the one for the advection phase and conveys similar information. The trend in timing plot in Figure 14b conforms with the prediction.

C. NAS CG

NAS-CG, which is part of NAS Parallel Benchmarks [4], uses a conjugate gradient method to approximate the smallest

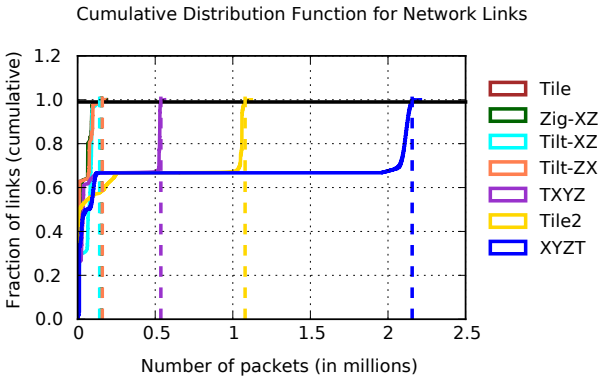


(a) CDF: Projection on x-axis predicts performance

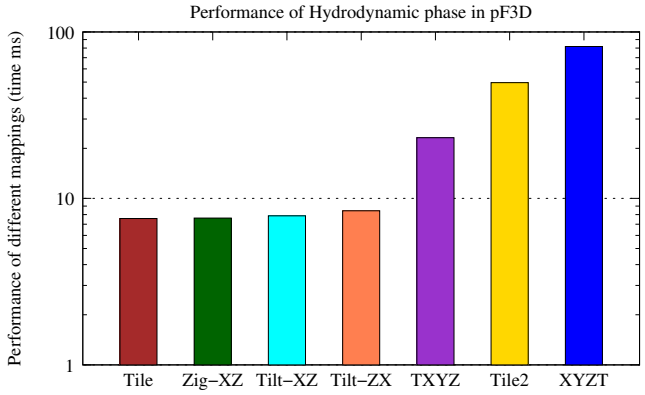


(b) Observed performance

Fig. 13: Advection phase: ghost-plane exchange is performed in Z-direction using `MPI_Isend`/ texttttMPI_Irecv. Plateau point of mappings with similar observed performance are very close, whereas for mappings with huge difference in performance, plateau points are far apart.



(a) CDF: Projection on x-axis predicts performance



(b) Observed performance

Fig. 14: Hydrodynamic phase: near-neighbor exchange in all six direction is done using `MPI_Isend`/`MPI_Irecv`; strong correlation is seen between x values of plateau point and the observed performance

Eigenvalue of a large sparse SPD matrix. It is used to test performance of irregular memory access and communication network of a system. It involves two important communication patterns - a reduction and a transpose exchange. The reduction phase involves multiple steps of exchanging data among pairs of processors, which is processed before the next step starts. Following the reduction, every processors exchanges data with a transpose processor. Both phases of the communication consume equal amount of time. Owing to the multi-phase structure, our analysis technique requires each of those phases to be analyzed separately. Given the lack of space, we only perform the analysis for the transpose phase, which is a good representative of all phases of communication in NAS-CG.

We ran NAS-CG using a customized torus FIFO mapping in which each processor FIFO is mapped to two torus FIFOs (Section IV). This setting was chosen because we obtain best application performance when each processor FIFO is mapped to two torus FIFOs. The following seven mappings have been used for presenting results for NAS-CG: two dimension ordering mappings - TXYZ and XYZT, two tiled mapping - Tile and Tile2, two tilted mapping - Tilt-XZ and Tilt2-XZ, and a zigzag based mapping - Zig2-XZ. In addition, we ran NAS-CG for 4 more mappings whose results are represented by one of the above mentioned mappings.

In Figure 15a, we present the CDF plots and the plateau points for the transpose exchange phase of NAS-CG. Note the relatively high volume of traffic on x-axis (in comparison to various phases of pF3D). The high volume increases the impact of every bin, and hence mappings whose plateau point lie in bins in proximity may have significant difference in performance. This is in contrast to some phases of pF3D with low communication volume, where a difference of few bins did not lead to significant impact on the performance.

Using CDFs in Figure 15a and correlating based on x values of plateau points, we expect Tilt2-XZ to have the best performance. It should be followed by Tilt-XZ and Zig2-

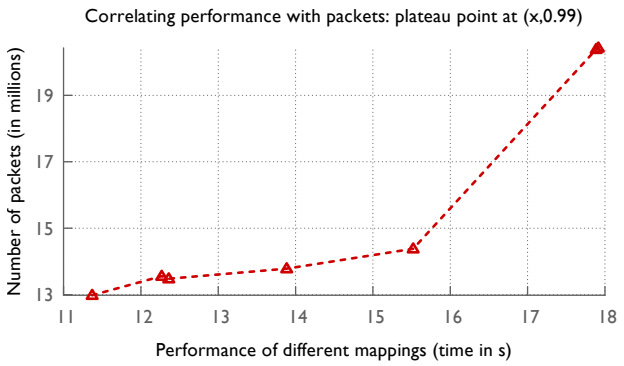


Fig. 16: NAS-CG: Correlating Performance with plateau point

XZ, which should have similar performance as x values of their plateau points are very close. Following them, we expect Tile2 and Tile to be the next best but with significant performance difference. Finally, XYZT and TXYZ, x values of whose plateau points is very high should have bad but similar performance.

The timing performance of this phase of NAS-CG is presented in Figure 15b with time measured in seconds. The first thing to note is the worst performing pair of XYZT and TXYZ. Moving left, Tile and Tile2 performs much better, and the trend conforms with the prediction. Following them, we find the pair of Zig2-XZ and Tilt-XZ with similar performance. Finally, Tilt2-XZ is the one with minimum execution time, as predicted by using x value of its plateau point as the correlation metric.

In Figure 16, we present a correlation plot for x values of plateau points and observed application performance for the given mappings. Earlier, in the section on choosing the best suited plateau point (Section VI), we showed a similar graph using 1.00 as y value in plateau points, which essentially implies using the number of packets on the most loaded link as the metric for predicting performance. We had found that such a metric did not yield a good correlation (Figure 9). In contrast, Figure 16 demonstrates that the x values of plateau point found using plateau points with $y = 0.99$ correlates well with the observed performance of NAS-CG.

VIII. Conclusion

Understanding and optimizing task mappings is essential to achieving good performance on current and future HPC platforms. While prior work has investigated this topic for latency-bound applications, few approaches have targeted bandwidth-bound applications, in particular, on torus networks. In contrast to task mappings for latency optimization, which typically target the minimization of hop counts between communication partners, mapping bandwidth-bound applications is more complicated, as it requires placements in a way such that we maximize the number of paths between communication partners, while still maintaining some form of locality.

In this work, we address this gap and provide a systematic approach to understanding the network performance for bandwidth-bound applications. Central to our approach is the plateau point metric, which provides a bound on the amount of traffic flowing through each link given a fraction of the number of links being considered. This definition helps omit outliers, e.g., in scenarios with a single heavily overloaded link. We show that our metric successfully characterizes the network traffic of different phases in two applications and enables us to make predictions about which mapping is best suited for a application or application phase. This gives us a powerful predictive capability and lays the foundation for future work on automatically generating optimal task mappings.

Acknowledgment

This research was supported in part by the Blue Waters: Leadership Petascale System project (which is supported by the NSF grant OCI 07-25070) and by the US Department of Energy under grant DOE DE-SC0001845. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-CONF-587533).

References

[1] T. Agarwal, A. Sharma, and L. V. Kalé. Topology-aware task mapping for reducing communication contention on large parallel machines. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium 2006*, April 2006.

[2] S. Alam, R. Barrett, M. Bast, M. R. Fahey, J. Kuehn, C. McCurdy, J. Rogers, P. Roth, R. Sankaran, J. S. Vetter, P. Worley, and W. Yu. Early evaluation of IBM Blue Gene/P. In *SC 08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12. IEEE Press, 2008.

[3] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman. Loggp: incorporating long messages into the logp modelone step closer towards a realistic model for parallel computation. In *Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*, SPAA '95, pages 95–105, New York, NY, USA, 1995. ACM.

[4] D. Bailey, E. Barszcz, L. Dagum, and H. Simon. NAS parallel benchmark results. In *Proc. Supercomputing*, Nov. 1992.

[5] P. Balaji, H. Naik, and N. Desai. Understanding network saturation behavior on large-scale blue gene/p systems. In *Proceedings of the 2009 15th International Conference on Parallel and Distributed Systems*, ICPADS '09, pages 586–593, Washington, DC, USA, 2009. IEEE Computer Society.

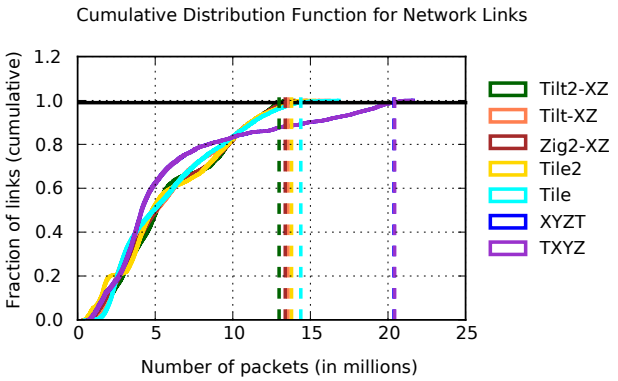
[6] A. Bhatele. *Automating Topology Aware Mapping for Supercomputers*. PhD thesis, Dept. of Computer Science, University of Illinois, August 2010. http://hdl.handle.net/2142/16578.

[7] A. Bhatele, E. Bohm, and L. V. Kale. Optimizing communication for charm++ applications by reducing network contention. *Concurrency and Computation: Practice and Experience*, 23(2):211–222, 2011.

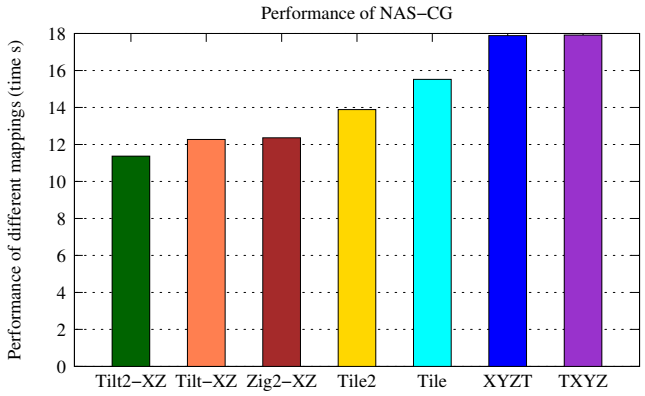
[8] A. Bhatele, T. Gamblin, S. H. Langer, P.-T. Bremer, E. W. Draeger, B. Hamann, K. E. Isaacs, A. G. Landge, J. A. Levine, V. Pascucci, M. Schulz, and C. H. Still. Mapping applications with collectives over sub-communicators on torus networks. In *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '12. IEEE Computer Society, Nov. 2012 (to appear). LLNL-CONF-556491.

[9] A. Chan, P. Balaji, W. Gropp, and R. Thakur. Communication analysis of parallel 3d fft for flat cartesian meshes on large blue gene systems. In P. Sadayappan, M. Parashar, R. Badrinath, and V. Prasanna, editors, *High Performance Computing - HiPC 2008*, volume 5374 of *Lecture Notes in Computer Science*, pages 350–364. Springer Berlin Heidelberg, 2008.

[10] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. Logp: Towards a realistic model of parallel computation. In *Fourth ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming PPOPP*, San Diego, CA, May 1993.



(a) CDF: Projection on x-axis predicts performance



(b) Observed performance

Fig. 15: NAS-CG: transpose exchange phase of NAS-CG; as the projection onto x-axis moves from left to right, the predicted performance is expected to be bad; we observe similar trend for predicted and observed performance.

[11] F. Ercal and J. Ramanujam and P. Sadayappan. Task allocation onto a hypercube by recursive mincut bipartitioning. In *Proceedings of the 3rd conference on Hypercube concurrent computers and applications*, pages 210–221. ACM Press, 1988.

[12] M. I. Frank, A. Agarwal, and M. K. Vernon. Lopc: modeling contention in parallel algorithms. In *Proceedings of the sixth ACM SIGPLAN symposium on Principles and practice of parallel programming*, PPOPP '97, pages 276–287, New York, NY, USA, 1997. ACM.

[13] T. Hoefler and M. Snir. Generic topology mapping strategies for large-scale parallel architectures. In *Proceedings of the international conference on Supercomputing*, ICS '11, pages 75–84, New York, NY, USA, 2011. ACM.

[14] C. L. Janssen, H. Adalsteinsson, S. Cranford, J. P. Kenny, A. Pinar, D. A. Evensky, and J. Mayo. A simulator for large-scale parallel computer architectures. *IJDST*, 1(2):57–73, 2010.

[15] C. Moritz and M. Frank. Logpg: Modeling network contention in message-passing programs. *Parallel and Distributed Systems, IEEE Transactions on*, 12(4):404 –415, apr 2001.

[16] C. H. Still, R. L. Berger, A. B. Langdon, D. E. Hinkel, L. J. Suter, and E. A. Williams. Filamentation and forward brillouin scatter of entire smoothed and aberrated laser beams. *Physics of Plasmas*, 7(5):2023, 2000.

[17] M. M. Tikir, M. A. Laurenzano, L. Carrington, and A. Snavely. Psins: An open source event tracer and execution simulator. *HPCMP Users Group Conference*, 0:444–449, 2009.

[18] J. S. Vetter, S. R. Alam, T. H. D. Jr., M. R. Fahey, P. C. Roth, and P. H. Worley. Early evaluation of the Cray XT3. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[19] G. Zheng, G. Kakulapati, and L. V. Kalé. Bigsim: A parallel simulator for performance prediction of extremely large parallel machines. In *18th International Parallel and Distributed Processing Symposium (IPDPS)*, page 78, Santa Fe, New Mexico, April 2004.